Documentazione

[1 Introduzione 2](#_Toc57900655)

[1.1 Informazioni sul progetto 2](#_Toc57900656)

[1.2 Abstract 2](#_Toc57900657)

[1.3 Scopo 2](#_Toc57900658)

[2 Analisi 3](#_Toc57900659)

[2.1 Analisi del dominio 3](#_Toc57900660)

[2.2 Analisi e specifica dei requisiti 3](#_Toc57900661)

[2.3 Use case 6](#_Toc57900662)

[2.4 Pianificazione 6](#_Toc57900663)

[2.5 Analisi dei mezzi 7](#_Toc57900664)

[2.5.1 Software 7](#_Toc57900665)

[2.5.2 Hardware 7](#_Toc57900666)

[3 Progettazione 8](#_Toc57900667)

[3.1 Design dell’architettura del sistema 8](#_Toc57900668)

[3.2 Design dei dati e database 8](#_Toc57900669)

[3.3 Design delle interfacce 8](#_Toc57900670)

[3.4 Design procedurale 8](#_Toc57900671)

[4 Implementazione 9](#_Toc57900672)

[5 Test 9](#_Toc57900673)

[5.1 Protocollo di test 9](#_Toc57900674)

[5.2 Risultati test 9](#_Toc57900675)

[5.3 Mancanze/limitazioni conosciute 9](#_Toc57900676)

[6 Consuntivo 10](#_Toc57900677)

[7 Conclusioni 11](#_Toc57900678)

[7.1 Sviluppi futuri 11](#_Toc57900679)

[7.2 Considerazioni personali 11](#_Toc57900680)

[8 Bibliografia 11](#_Toc57900681)

[8.1 Sitografia 11](#_Toc57900682)

[9 Allegati 11](#_Toc57900683)

# Introduzione

## Informazioni sul progetto

• Mandante: Geo Petrini.

• Partecipanti: Daniele Cereghetti.

• Tempo a disposizione: 1 settembre 2020 – 23 dicembre 2020

• Classe: I3AC.

• Progetto: LanSpeedTester

## Abstract

## Scopo

Lo scopo del progetto è lo sviluppo di un’applicazione per il test della velocità di trasferimento dati all’interno di una rete locale. Il software sarà diviso in due parti, la parte server e la parte client: il server si occupa di ricevere i dati e ritornarli, invece il client oltre a inviare e ricevere i dati, si preoccupa di cronometrare il tempo che passa tra l’invio e la ricevuta, alla fine del test il client mostra la statistica dei risultati ottenuti con la media dei tempi, il numero di pacchetti ricevuti e persi e la velocità della LAN (es. 10Mb/s).

# Analisi

## Analisi del dominio

Questo programma verrà utilizzato in ambito aziendale per verificare la velocità della rete (LAN) tra due computer, oppure tra un server fisico (server con un DB, un FTP, un NAS, …) e un pc.

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: 1** | |
| **Nome** | SW con cambio ruolo (Server o Client) |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: 2** | |
| **Nome** | Il Software non deve essere installato e non salva niente sul PC |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: 3** | |
| **Nome** | Utilizza un Linguaggio multi-piattaforma |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Utilizzo il linguaggio Java |

|  |  |
| --- | --- |
| **ID: 4** | |
| **Nome** | L’utente del server sceglie la porta d’ascolto |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: 5** | |
| **Nome** | Il server mostra lo stato del client |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: 6** | |
| **Nome** | Il server apre porta e accetta connessioni del client |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | questa cosa server anche per l'utilizzo normale, ovvero che per far funzionare correttamente il programma, prima si fa partire il server e successivamente quando il server è "online" si fa "accende il client |

|  |  |
| --- | --- |
| **ID: 7** | |
| **Nome** | Programma da terminale (CLI) |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: 8** | |
| **Nome** | Server e client possono sfruttare il multithreading |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Il server deve accettare una o più connessione in entrata, il client deve poter aprire più connessioni verso lo stesso server; entrambi sulla stessa porta o su porte diverse |

|  |  |
| --- | --- |
| **ID: 9** | |
| **Nome** | L’utente del client deve scegliere il peso di ogni pacchetto da inviare al server |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: 10** | |
| **Nome** | L’utente del client deve scegliere il numero di connessioni da fare al server |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: 11** | |
| **Nome** | Il client deve mostrare la statistica dei risultati |
| **Priorità** | 3 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: 12** | |
| **Nome** | Misurazione dei tempi al millisecondo |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** |  |

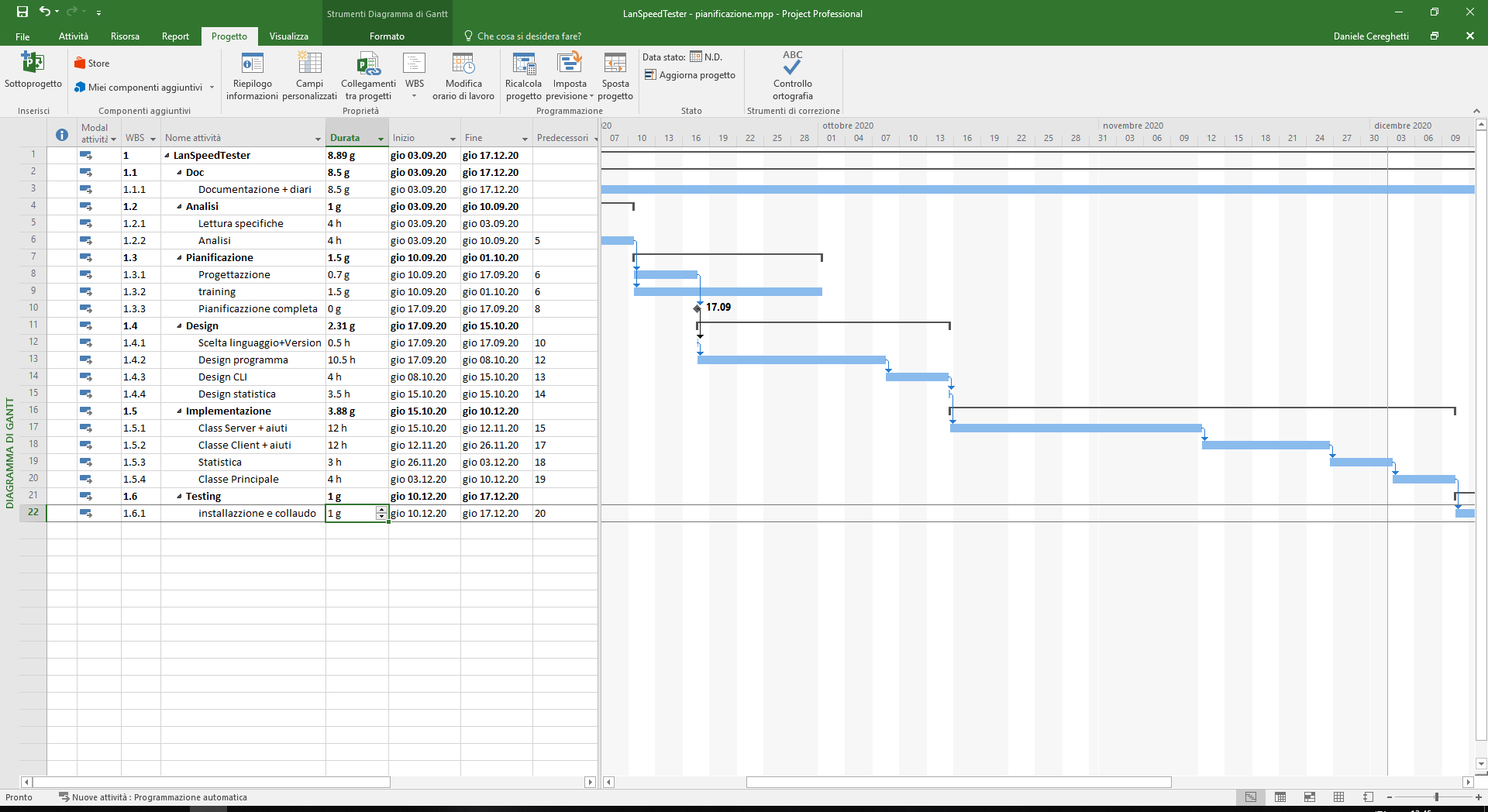
|  |  |
| --- | --- |
| **ID: 13** | |
| **Nome** | Riportare nelle statistiche se è stato sfruttato il multithreading o no |
| **Priorità** | 3 |
| **Versione** | 1.0 |
| **Note** |  |

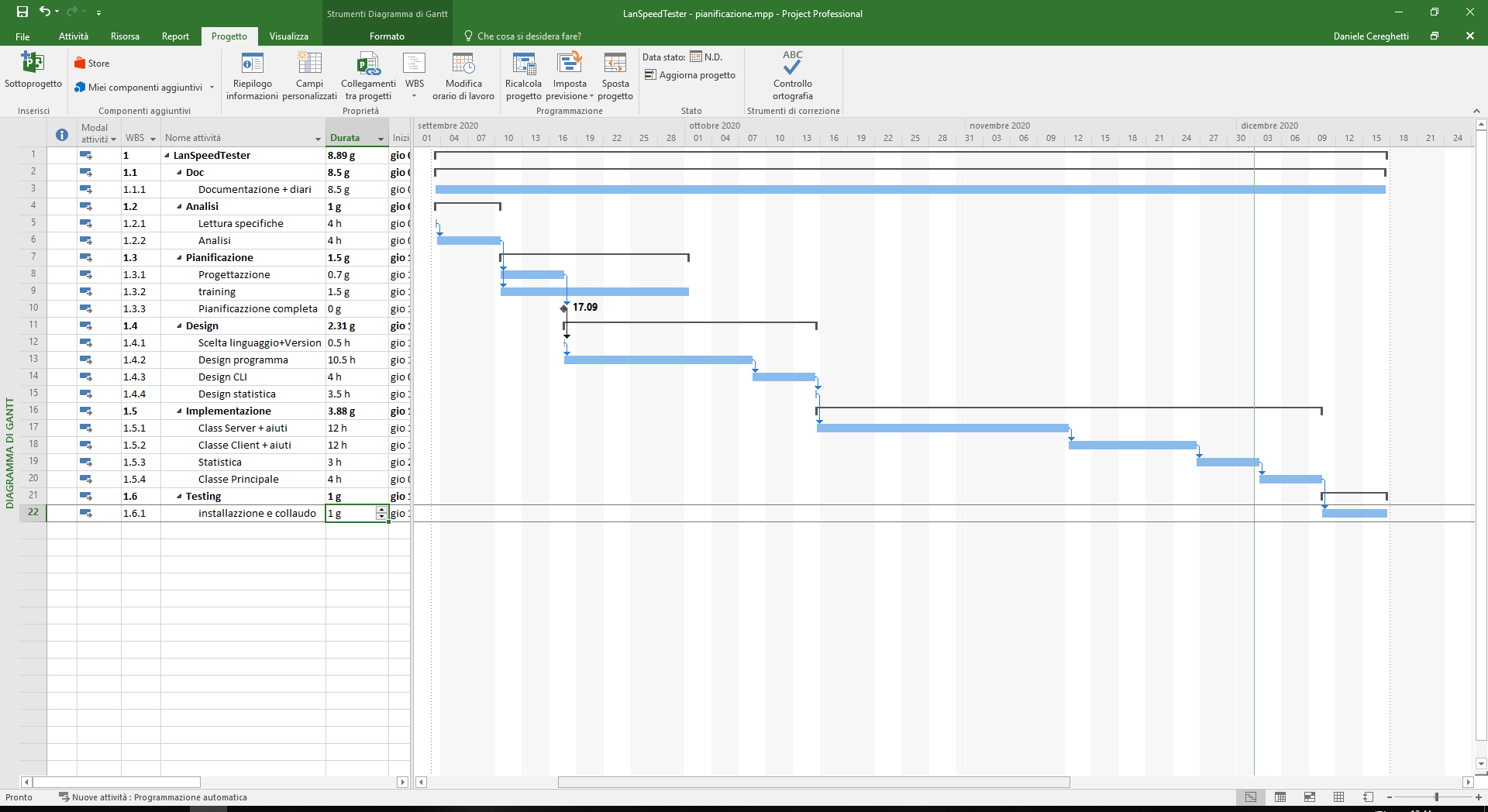
## Use case

**Attenzione** questo programma lavora esclusivamente su reti con IPv4, non supporta l’IPv6.

Prendo i due computer sui quali voglio eseguire il test, su uno dei due si fa partire la funzionalità del server, si inserisce la porta d’ascolto, da quel momento il server è in ascolto e verrà visualizzato l’indirizzo IP del computer in questione e la porta d’ascolto. Successivamente sull’altro computer si esegue la funzione client, si inserisce il numero di connessioni da fare per il test, il peso di ogni pacchetto da inviare, l’IP dell’altro pc e la sua porta d’ascolto. Se tutti i dati inseriti nel “client” sono corretti sul server mostrerà che il client è connesso e i dati per il test, successivamente mostrerà che il test ha inizio.

## Pianificazione





## Analisi dei mezzi

Una macchina con cui scrivere un progetto, una macchina virtuale per fare i test e una connessione a internet per le ricerche sul linguaggio, disegno architettura, utilizzo di GitHub, …

### Software

Ide per Java:

* NetBeans IDE 8.2 o superiore

Versione di java (JDK):

* java version "1.8.0\_172" o superiore

Gant:

* Microsoft Project 2016

Design architettura:

* <https://app.diagrams.net/>

### Hardware

* CPU 🡪 intel core i7-7700 @ 3.6 GHz
* RAM 🡪 16 GB
* SSD 🡪 500 GB
* Architettura 🡪 64 bit

# Progettazione

## Design dell’architettura del sistema

## Design dei dati e database

## Design delle interfacce

## Design procedurale

# Implementazione

All’inizio avevo pensato che il server aprisse una sola porta per il test, e quando arrivava il dato c’era un listener che si occupava di smistare le cose, però mi ha dato abbastanza problemi questa cosa quindi ho deciso che il client inviava al server le sue porte libere e il server le sue, così tutte e due hanno le rispettive porte dell’altro, e questo metodo mi ha dato un solo problema, ovvero sbagliavo a dargli la porta sulla quale aprire la connessione (per entrambi).

# Test

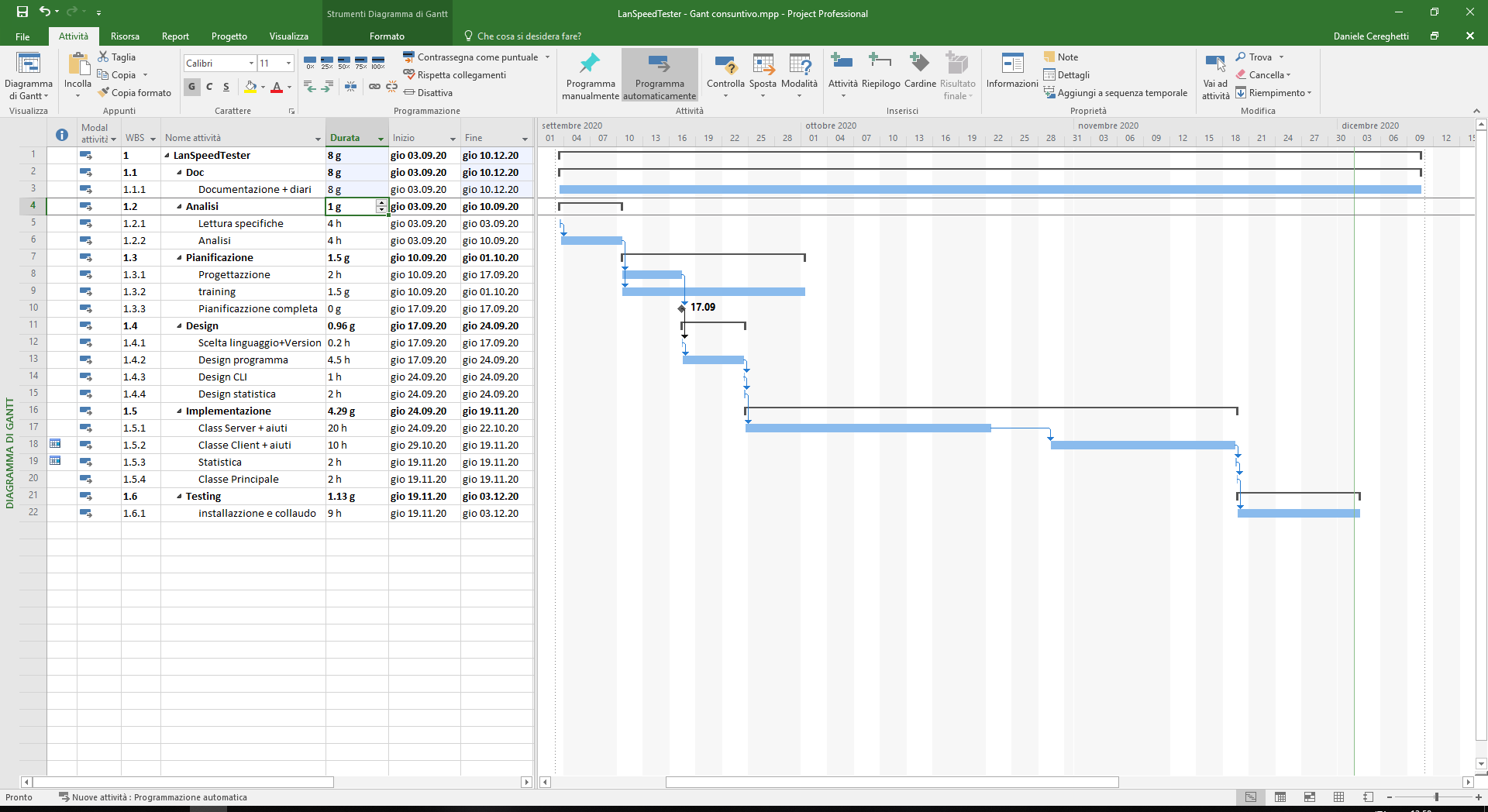
## Protocollo di test

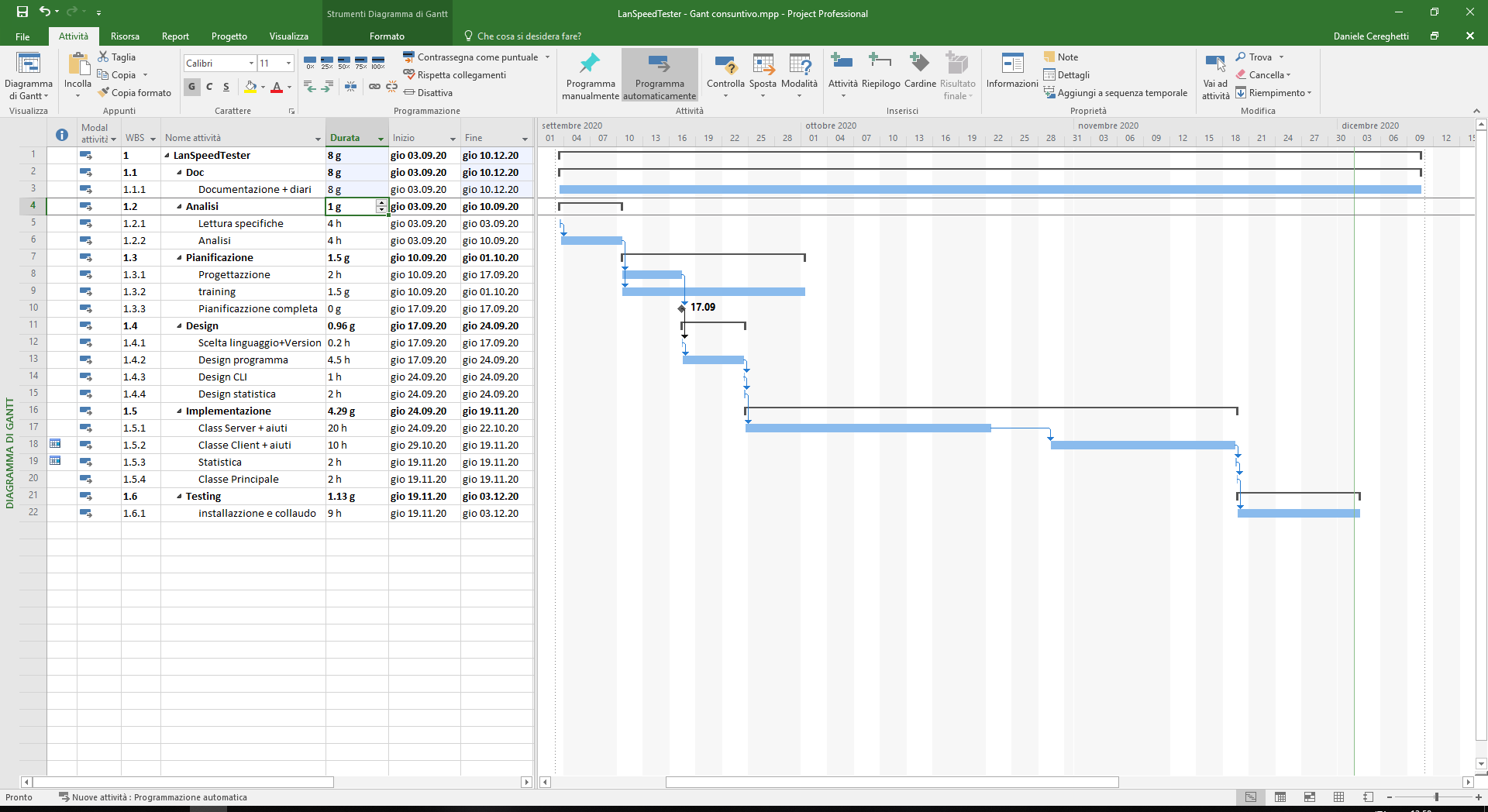
|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-012 | **Nome:** | Import a card with KIC, KID and KIK keys, but not shown with the GUI |
| **Descrizione:** | Import a card with KIC, KID and KIK keys with no obfuscation, but not shown with the GUI | | |
| **Prerequisiti:** | Store on local PC: Profile\_1.2.001.xml (appendix n\_n) and Cards\_1.2.001.txt (appendix n\_n).  PIN (OTA\_VIEW\_PIN\_PUK\_KEY) and ADM (OTA\_VIEW\_ADM\_KEY) user right not set. | | |
| **Procedura:** | 1. Go to “Cards manager” menu,  in main page click “Import Profiles” link, Select the “1.2.001.xml” file, Import the Profile 2. Go to “Cards manager” menu,  in main page click “Import Cards” link, Select the “1.2.001.txt” file, Delete the cards,  Select the “1.2.001.txt” file, Import the cards 3. Research the “41795924770” Card, Click the imsi card link Check the card details 4. Execute the SQL: SELECT imsi, dir, keyset, cntr, rawtohex(kickey), rawtohex(kidkey), rawtohex(kikkey), rawtohex(chv), rawtohex(dap)FROM otacardkey a where imsi='340041795924770' ORDER BY keyset; | | |
| **Risultati attesi:** | Keys visible in the DB (OtaCardKey) but not visible in the GUI (Card details) | | |

## Risultati test

## Mancanze/limitazioni conosciute

# Consuntivo





Confrontando il gantt consuntivo con quello di pianificazione si nota che fino alla pianificazione sono stato al passo con la pianificazione, ma non capisco perché il design è durato molto meno di quello che avevo pensato, e mi ha anche permesso di avere un certo anticipo. Ma nonostante avessi preso un certo vantaggio pensavo che fosse stato più facile scrivere il codice, invece è durato poco di più ma ho avuto parecchi problemi per la parte server, il client invece resta nel suo tempo ideale, invece la cosa che mi ha stupito in positivo è stata la velocità con cui ho fatto la statistica e la classe principale. La parte di testing invece è durata più di quello previsto perché alcune formule davano gli errori e ci ho messo un po’ a trovare una soluzione.

# Conclusioni

## Sviluppi futuri

Si potrebbe far supportare l’IPv6, migliorare il metodo di invio dei dati, perché adesso in ram si allocano (nei casi estremi) 100 – 130 MB, quindi si potrebbe memorizzare un byte solo, e inviarlo N volte così da inviare in totale il numero di byte richiesti dall’utente.

## Considerazioni personali

Questo progetto mi ha aiutato a capire che sono bravino nella logica e che ho bisogno di tempo per capirla.

Mi è piaciuto anche perché ho imparato tante cose nuove.

# Bibliografia

## Sitografia

1. <https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/net/package-summary.html>

Java.net (java SE 12 & JDK 12), 22.10.2020

1. <https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/net/Socket.html>

Socket (java SE 12 & JDK 12), 22.10.2020

1. <https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/net/ServerSocket.html>

ServerSocket (java SE 12 & JDK 12), 22.10.2020

1. <https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/net/DatagramSocket.html>

DatagramSocket (java SE 12 & JDK 12), 15.10.2020

1. <https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/net/DatagramPacket.html>

DatagramPacket (java SE 12 & JDK 12), 15.10.2020

1. <https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/lang/Thread.html>

Thread (java SE 12 & JDK 12), 22.10.2020

1. <https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/net/Inet4Address.html>

Inet4Address (java SE 12 & JDK 12), 8.10.2020

1. <https://stackoverflow.com/questions/6916398/communicating-between-two-threads>

Forum per Queue Java, 29.10.2020

1. <https://www.codejava.net/java-se/networking/java-socket-server-examples-tcp-ip>

Esempio socket tcp, 22.10.2020

1. <https://www.tutorialspoint.com/javaexamples/net_multisoc.htm>

Esempio socket tcp, 22.10.2020

1. <https://www.codejava.net/java-se/networking/java-udp-client-server-program-example>

Esempio socket udp, 15.10.2020

# Allegati

Per i diari, codici sorgente e prodotto finale 🡪 <https://github.com/Daniele-Cereghetti/LanSpeedTester>